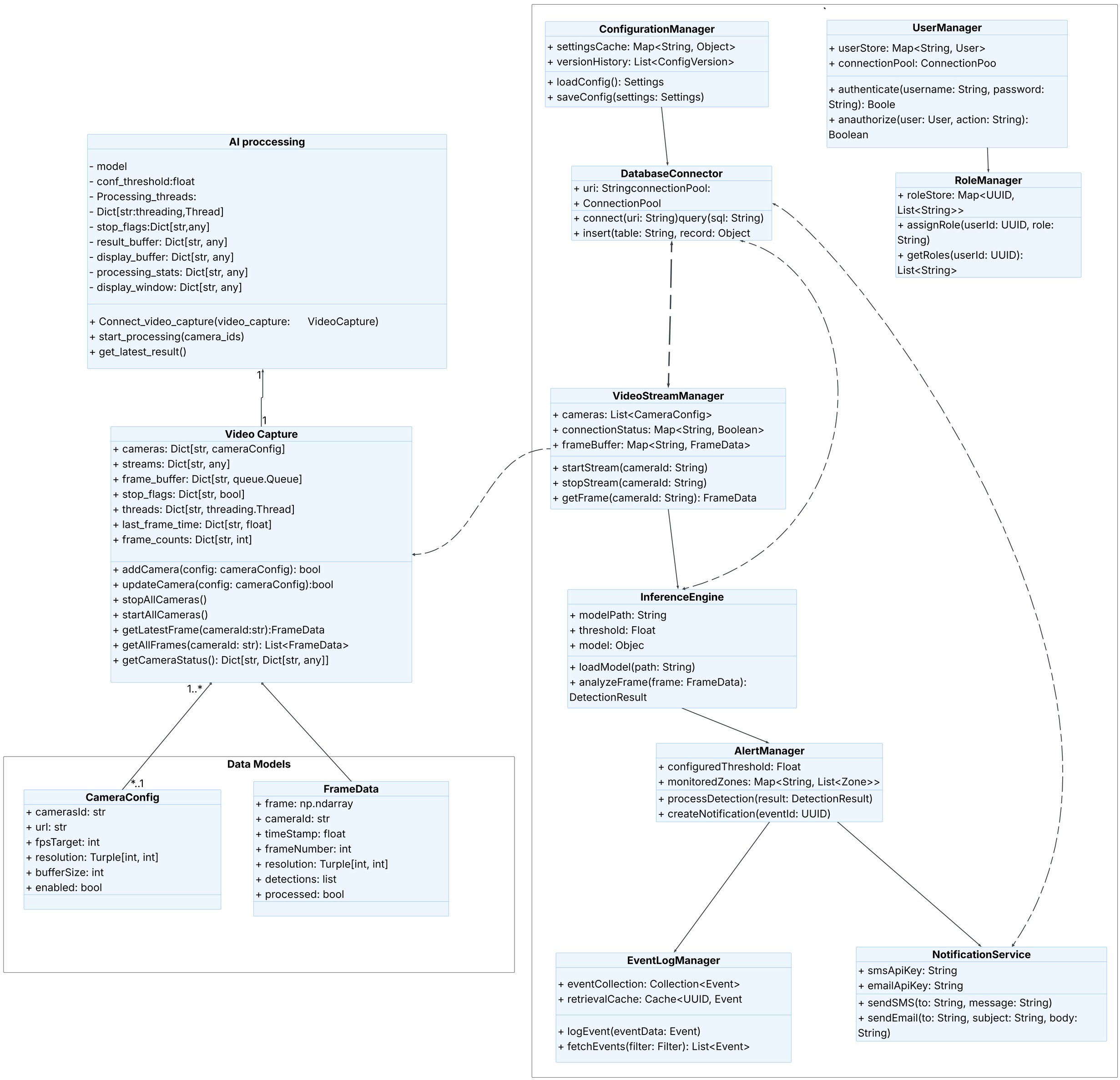
AI proccessing
- model
- conf_threshold:float
- Processing_threads:
- Dict[str:threading,Thread]
- stop_flags:Dict[str,any]
- result_buffer: Dict[str, any]
- display_buffer: Dict[str, any]
- processing_stats: Dict[str, any]
- display_window: Dict[str, any]
+ Connect_video_capture(video_capture: VideoCapture)
+ start_processing(camera_ids)
+ get_latest_result()
1
1
Video Capture
+ cameras: Dict[str, cameraConfig]
+ streams: Dict[str, any]
+ frame_buffer: Dict[str, queue.Queue]
+ stop_flags: Dict[str, bool]
+ threads: Dict[str, threading.Thread]
+ last_frame_time: Dict[str, float]
+ frame_counts: Dict[str, int]
+ addCamera(config: cameraConfig): bool
+ updateCamera(config: cameraConfig):bool
+ stopAllCameras()
+ startAllCameras()
+ getLatestFrame(cameraId:str):FrameData
+ getAllFrames(cameraId: str): List<FrameData>
+ getCameraStatus(): Dict[str, Dict[str, any]]
1..*
*..1
Data Models
CameraConfig
+ camerasId: str
+ url: str
+ fpsTarget: int
+ resolution: Turple[int, int]
+ bufferSize: int
+ enabled: bool
FrameData
+ frame: np.ndarray
+ cameraId: str
+ timeStamp: float
+ frameNumber: int
+ resolution: Turple[int, int]
+ detections: list
+ processed: bool
ConfigurationManager
+ settingsCache: Map<String, Object>
+ versionHistory: List<ConfigVersion>
+ loadConfig(): Settings
+ saveConfig(settings: Settings)
UserManager
+ userStore: Map<String, User>
+ connectionPool: ConnectionPoo
+ authenticate(username: String, password: String): Boole
+ anauthorize(user: User, action: String): Boolean
DatabaseConnector
+ uri: StringconnectionPool:
+ ConnectionPool
+ connect(uri: String)query(sql: String)
+ insert(table: String, record: Object
RoleManager
+ roleStore: Map<UUID, List<String>>
+ assignRole(userId: UUID, role: String)
+ getRoles(userId: UUID): List<String>
VideoStreamManager
+ cameras: List<CameraConfig>
+ connectionStatus: Map<String, Boolean>
+ frameBuffer: Map<String, FrameData>
+ startStream(cameraId: String)
+ stopStream(cameraId: String)
+ getFrame(cameraId: String): FrameData
InferenceEngine
+ modelPath: String
+ threshold: Float
+ model: Objec
+ loadModel(path: String)
+ analyzeFrame(frame: FrameData): DetectionResult
AlertManager
+ configuredThreshold: Float
+ monitoredZones: Map<String, List<Zone>>
+ processDetection(result: DetectionResult)
+ createNotification(eventId: UUID)
EventLogManager
+ eventCollection: Collection<Event>
+ retrievalCache: Cache<UUID, Event
+ logEvent(eventData: Event)
+ fetchEvents(filter: Filter): List<Event>
NotificationService
+ smsApiKey: String
+ emailApiKey: String
+ sendSMS(to: String, message: String)
+ sendEmail(to: String, subject: String, body: String)